***Запропоновано розширення функцій системи оцінки ідентичності програмного коду при вивченні та практичному застосуванні мов програмування в процесі навчання, шляхом використання сформованих репозиторіїв розподіленої системи контролю версій***

***Ключові слова: контроль версій, навчальний процес, плагіат***

***Предложено расширение функций системы оценки идентичности программного кода при изучении и практическом применении языков программирования в процессе обучения, путем использования сформированных репозиториев распределенной системы контроля версий***

***Ключевые слова: контроль версий, учебный процесс, плагиат***

***The expanding functions of the estimation system of identity program code in the study and practical application of programming languages in the learning process through the use of generated repositories distributed version control system are proposed***

***Keywords: version control, learning process, plagiarism***

УДК 004.43, 004.9

# МОДЕЛЬ ОЦІНКИ ПЛАГІАТУ ПРОГРАМНОГО КОДУ НА ОСНОВІ СИСТЕМИ КОНТРОЛЮ ВЕРСІЙ

**Г.Г. Киричек**
Кандидат технічних наук, доцент
Кафедра комп'ютерних систем та мереж
Запорізький національний технічний університет
вул. Жуковського, 64, м. Запоріжжя, 69063
Контактний тел.: (061) 787-56-31, 050-484-85-79
E-mail: kirichek@zntu.edu.ua; kirgal08@mail.ru

**О.О. Киричек**
Національний технічний університет України
"Київський політехнічний інститут"
пр. Перемоги, 37, м. Київ, 03056
Контактний тел.: 095-779-44-89
E-mail: sashakirichek@ukr.net

## 1. Вступ

Одним з актуальних питань у вищих навчальних закладах (ВНЗ) є забезпечення ефективності вивчення дисциплін з програмування [1]. Можливість легкого копіювання інформації, яка представлена у електронному вигляді, при виконанні індивідуальних завдань з розробки програмного коду породила багато проблем, які пов'язані з правильністю оцінки індивідуальних робіт студентів. У зв'язку з цим виникла необхідність визначення повторного використання програмного коду у творчих проектах студентів, при вивченні дисциплін з програмування, з використанням системи оцінки ідентичності програмного коду [2].

Крім того останнім часом активну популярність набирають розподілені системи контролю версій. Найбільш поширеними з таких є Subversion, Git та Mercurial. Знання подібних систем підвищує затребуваність IT фахівців на ринку праці, покращує продуктивність розробників та полегшує рішення щоденних завдань. Саме передача знань є вирішальною у процесі експорту-імпорту технологій [3].

Під час контролю програмного коду, який розроблено студентами при вивченні дисциплін з програмування, викладачі стикаються з проблемами синхронізації та ведення історії файлів, вирішення яких можливе при застосуванні однієї з систем керування версіями. Тому розширення функцій системи оцінки ідентичності програмного коду, при вивченні та практичному застосуванні мов програмування в процесі навчання з використанням сформованих репозиторіїв розподіленої системи контролю версій, є актуальним завданням, яке потребує проведення додаткових досліджень.

## 2. Проведення аналізу та вибір системи контролю версій

Вирішуючи завдання розширення функцій системи оцінки ідентичності програмного коду з використанням сформованих репозиторіїв розподіленої системи контролю версій, розглянемо системи керування контролю версій з точки зору використання їх у навчальному процесі.

Subversion - централізована система, яка зберігає дані в єдиному сховищі. При збереженні нових версій використовується дельта-компресія, тобто система знаходить відмінності нової версії від попередньої і записує тільки їх, уникаючи непотрібного дублювання даних. Сховище може розташовуватися на локальному диску або на мережевому сервері. До локального сховища клієнт Subversion звертається безпосередньо, а для доступу до віддаленого сервера може використовуватися власний мережевий протокол або стандартний протокол WebDAV, який підтримується за допомогою спеціального модуля для веб-сервера Apache.

При використанні цієї системи студенти мали б можливість копіювати файли зі сховища, створювати локальні робочі копії, потім модифікувати їх і

публікувати зміни в сховищі. До сховища одночасно можуть звертатися декілька студентів та викладачів.

При доступі за допомогою WebDAV підтримується прозоре управління версіями. Тому коли будь-який студент відкриє для запису, а потім збереже файл, який розташовано на мережевому ресурсі, автоматично створиться нова версія цього файлу, що полегшить здійснення контролю розробки програмного коду викладачем.

Git - розподілена система керування версіями файлів та спільної роботи. Система спроектована як набір програм, спеціально розроблених з урахуванням їх використання в скриптах. Це дозволяє зручно створювати спеціалізовані системи контролю версій на базі Git або користувальницькі інтерфейси.

Git підтримує швидке розділення та злиття версій, містить можливості для візуалізації і навігації за нелінійною історією розробки. При використанні в процесі навчання вона надасть кожному викладачу копію всієї історії розробки, що дозволить здійснити контроль змін, які копіюються з одного сховища в інше.

Віддалений доступ до репозиторіїв Git забезпечується GIT-демоном, SSH-або HTTP-сервером. TCP-сервіс GIT-демон входить в дистрибутив Git і є разом з SSH найбільш поширеним і надійним методом доступу. Метод доступу по протоколу HTTP, незважаючи на ряд обмежень, дуже популярний в контрольованих мережах. Він дозволяє використовувати існуючі конфігурації мережевих фільтрів.

Mercurial - кроссплатформена розподілена система керування версіями. Вона розроблена для ефективної роботи з дуже великими репозиторіями програмного коду та, у першу чергу, є консольною програмою. Система Mercurial написана на Python, хоча чутливі до продуктивності частини виконані в якості Python-розширень на мові програмування C. Репозиторії Mercurial управляються за допомогою утиліти командного рядка hg [4].

Поряд з традиційними можливостями інших систем контролю версій, Mercurial підтримує повністю децентралізовану роботу (відсутнє поняття основного сховища коду), розгалуження (можливо вести кілька гілок одного проекту і копіювати зміни між гілками), злиття репозиторіїв (цим досягається «розподіленість» роботи). Підтримується обмін даними між репозиторіями через HTTP / HTTPS, SSH1 і вручну за допомогою упакованих наборів внесених змін.

Також Mercurial використовує SHA1-хеши для ідентифікації ревізій і дозволяє присвоювати окремим ревізіям індивідуальні мітки.

Дослідження проведені Google підтверджують, що використання систем Git та Mercurial стає дуже поширеним, в той час як Svn втрачає позиції домінуючої системи контролю версій (рис. 1).

Також у зв'язку з тим, що Subversion є централізованою системою, студенту при роботі буде потрібен постійно працюючий svn-сервер для синхронізації, що є дуже незручним. Тому вибір доцільно проводити між Mercurial та Git, як розподілених систем, шляхом порівняння за декількома критеріями та їх відповідності завданням, які пов'язані з навчальним процесом.

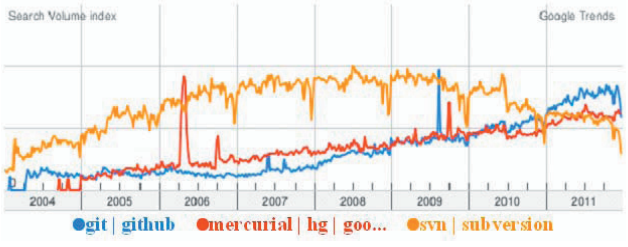


Рис. 1. Аналіз використання систем

В Git крива навчання більше крута, ніж в Mercurial у зв'язку з безліччю факторів. В Git більше команд і опцій, але їхня кількість завелика для користувачів. А документація Mercurial виглядає більше закінченої та простою для розуміння. Термінологія Mercurial ближче до Subversion і CVS, що робить більш простим освоєння цієї системи.

Git вимагає періодичного обслуговування репозиторіїв, а для Mercurial подібних дій робити не потрібно. Git - інструмент дуже потужний, і зробить практично все, що ви попросите. Це означає, що Git може видалити історію. Репозиторії ж Mercurial структуровані як постійно зростаюча колекція недоторканних об'єктів.

Git явно не відслідковує перейменування або копіювання файлів. Його команди шукають файли, ідентичні тим, які з'являлися раніше в історії репозиторія, і роблять висновок про перейменування або копіювання. Mercurial надає конкретні команди з перейменування та копіювання і зберігає ці дії як історії файлу.

Git складається з великої кількості shell скриптів і unix команд, які реалізовані на C. Mercurial написано на Python з використанням API. Це дозволяє стороннім розробникам нарощувати Mercurial за допомогою своїх Python модулів.

В Git, кожний репозиторій має свій власний простір імен гілок, і користувачі налаштовують зв'язки між локальними та віддаленими просторами імен. Mercurial має єдиний простір імен гілок, який використовується всіма репозиторіями [5].

З погляду можливостей Git більш потужний, але за рахунок складності використання. З погляду реалізації основних дій, Mercurial має перевагу завдяки ефективному протоколу поверх HTTP.

## 3. Реалізація системи

Після проведення аналізу даних систем, в якості підтримки репозиторіїв при виконанні практичних та індивідуальних робіт з програмування, була обрана розподілена система контролю версій Mercurial, як найбільш проста у використанні, загальнодоступна і задовольняюча всім програмним та технічним характеристикам підтримки навчального процесу.

Реалізація запропонованого рішення виконана заміною існуючої моделі роботи з проектами в системі оцінки ідентичності програмного коду [2,6], шляхом додавання додаткових модулів та класів.

В процесі розробки в системі додатково створено клас Man, який зберігає лише інформацію про користувача та код запрошення. Код надається студенту і

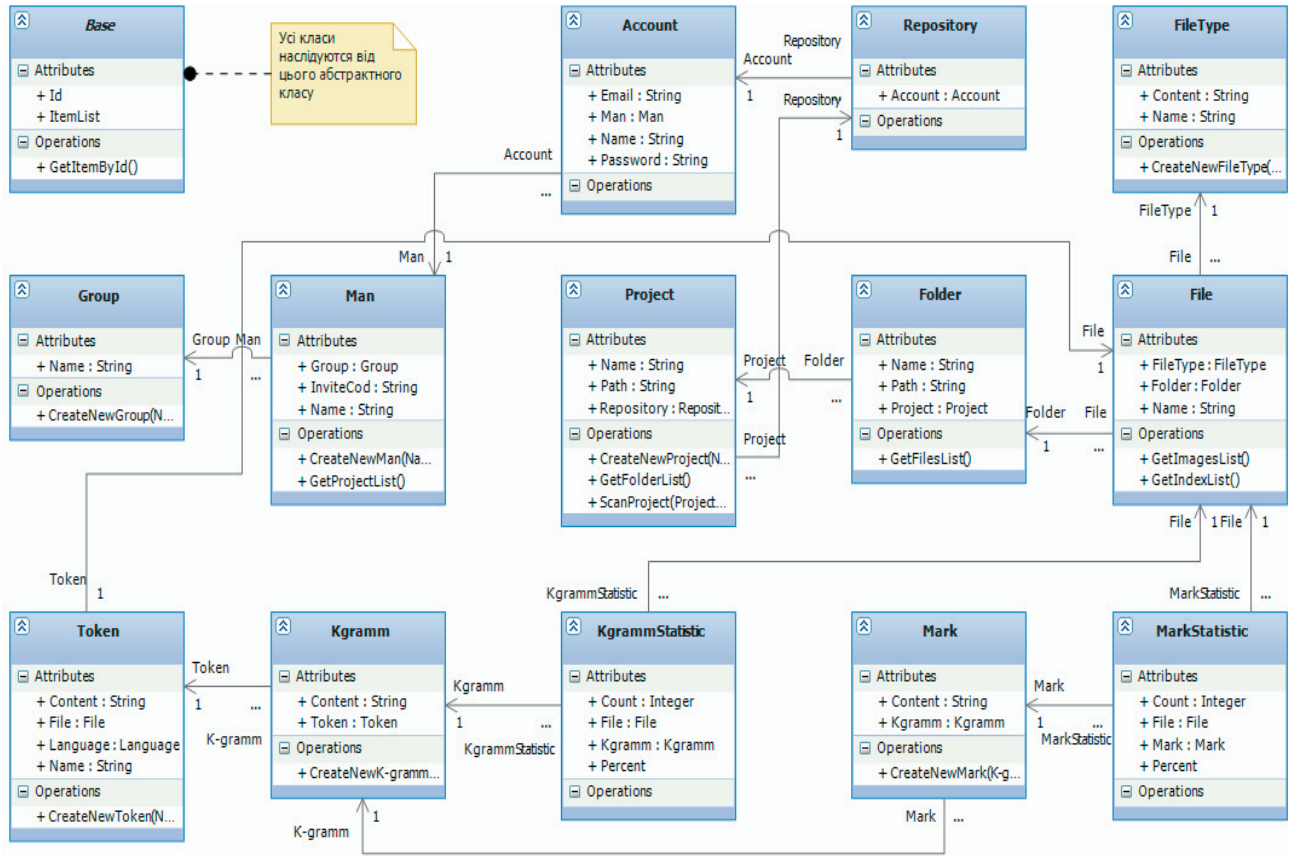


Рис. 2. Діаграма класів системи

використовується зареєстрованим користувачем для зв'язку свого Account та класу Man [7], шляхом введення цього коду при активації в системі.

Модуль Repository підтримує механізм роботи з розподіленою системою контролю версій Mercurial та розробленим алгоритмом розпізнавання плагіату при проведенні контролю та оцінки викладачем програмного коду в процесі навчання (рис. 2).

Локальні адміністратори (далі старости груп) реєструються в системі, використовуючи запрошення викладача. Далі вони вносять інформацію про студентів групи та надсилають їм запрошення до активації в системі. Студенти груп, використовуючи код запрошення, активуються для роботи в системі за допомогою e-mail та очікують підтвердження від старост.

Після підтвердження активації студент одержує доступ до системи керування репозиторіями своїх файлів. Тепер вони мають можливість додавати та видаляти свої репозиторії, відкривати до них доступ іншим студентам, старостам і викладачам, а також робити дії по адмініструванню режимів доступу користувачів та ір-адрес, з яких відбуваються операції редагування та додавання даних у кожен із репозиторіїв.

При закінченні виконання індивідуального завдання або на вимогу викладача студент надсилає викладачеві посилання на репозиторій, який містить вихідні коди розробленої програми для її перевірки.

Викладач виконує функцію перевірки репозиторія на плагіат і одержує інформацію про автентичність розробки (рис. 3).

Перевірка програмного коду здійснюється як існуючими способами, так і новим, який заснований на інформації про репозиторії та є доступним із системи контролю версій:

```
$ hg log -r1
changeset: 1:107e439be01c
tag: mytag
user: Bryan O'Sullivan <bos@serpentine.com>
date: Tue May 05 06:55:46 2012 +0000
summary: added line to end of <<hello>> file.
```

Опираючись на наведений вище фрагмент програмного коду робимо висновок, що синтаксично аналізуючи результат виконання команд можна отримати всю необхідну інформацію про дату, час, автора та характер проведених змін для кожного файлу.

При аналізі бази розміщених проектів можна за рядом критеріїв визначити ідентичність розроблених студентами програм при виконанні практичних та індивідуальних завдань, що веде до підвищення ефективності навчального процесу.

Розподілена система дозволяє гнучко допрацьовувати обмеження за розмірами репозиторіїв та кількістю збережених даних в об'ємі, який виділяється на одного студента.

Все це надає можливість адаптувати систему під доступні обсяги даних.

В результаті проведених досліджень отримано модель оцінки плагіату програмного коду на основі системи програмної роботи з репозиторіями розподіленої системи контролю версій Mercurial.

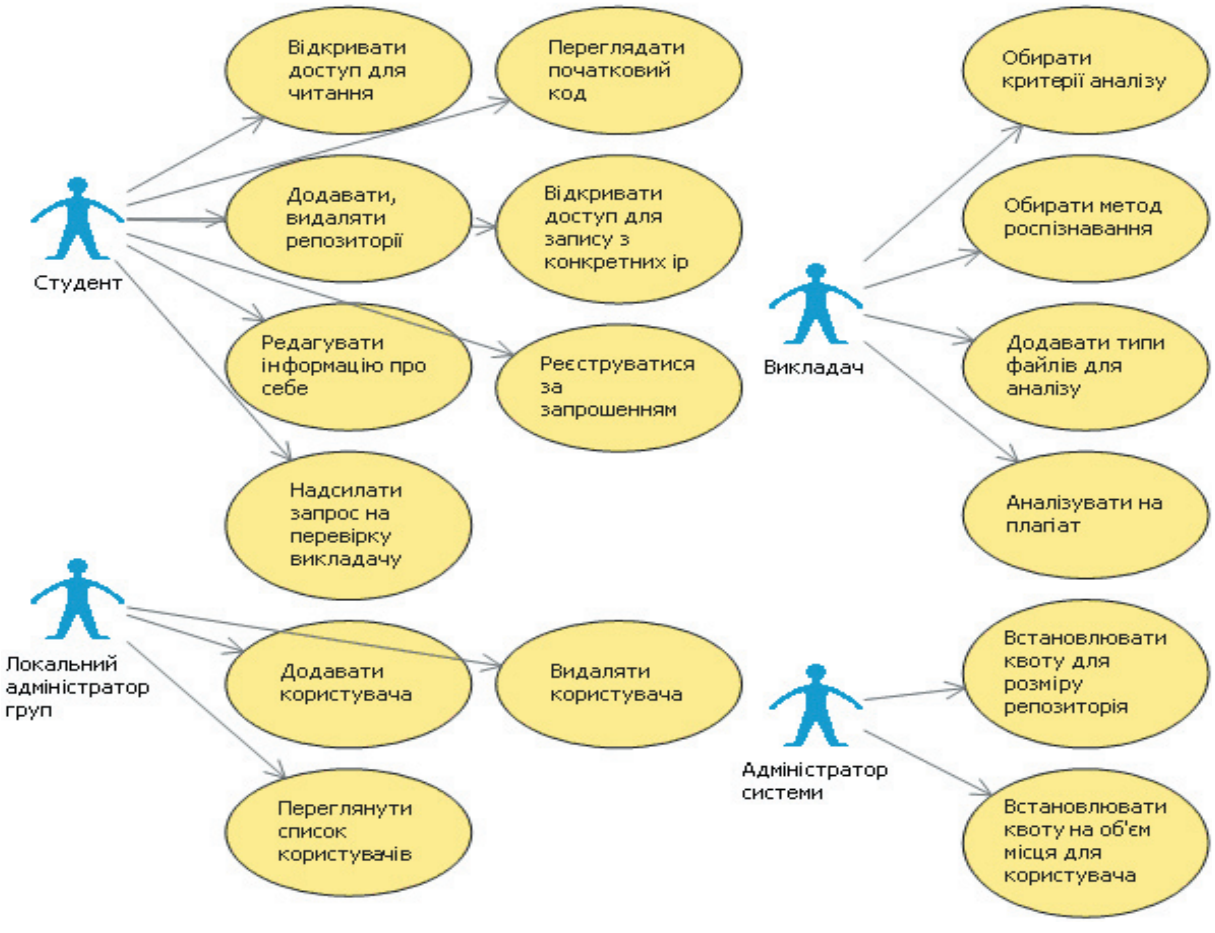


Рис. 3. Діаграма варіантів використання системи

## 4. Висновки

В роботі послідовно розглянуто характеристики розподілених систем контролю версій, їх архітектура та основні особливості з точки зору використання при формуванні репозиторіїв студентських розробок програмного коду в процесі вивчення дисциплін з програмування.

Проведено аналіз та обрано задовольняючу характеристикам моделі розподілену систему контролю версій. Розроблено модель оцінки плагіату програмного коду на основі використання розподіленої системи контролю версій Mercurial.

При проведенні подальших досліджень планується розробка, на основі запропонованої моделі, програми для платформи Windows 8 та впровадження модуля підтримки мобільної платформи Windows Phone 7.5 для зручного перегляду викладачем статистичних даних про наявність плагіату програмного коду у проектах студентів.

Література

1. Биков В.Ю. Моделі організаційних систем відкритої освіти: Монографія [Текст] / В.Ю. Биков. – К.: Атака, 2008. – 684 с.: іл.
2. Киричек Г.Г. Модель системи оцінки ідентичності програмного коду [Текст] / Г.Г. Киричек, О.О. Киричек // Науковий вісник Чернівецького національного університету. Комп'ютерні системи та компоненти. – Т. 2. – Вип.3. – 2011. – С. 14-21.
3. Li-Hua Richard. From technology transfer to knowledge transfer – a study of international joint venture projects in China [Текст] / Richard Li-Hua. – Newcastle Business School: University of Northumbria, 2008. – 23p.
4. Robbins Jason. Analysis of Git and Mercurial [Заглавие с экрана] / Jason Robbins: Режим доступу: http://code.google.com/p/support/wiki/DVCSAnalysis.
5. O'Sullivan Bryan. Distributed revision control with Mercurial [Заглавие с экрана] / Bryan O'Sullivan. – 2007. – 211p.: Режим доступу: http://hgbook.red-bean.com/hgbook.html.
6. Stepanova E.B., Krivtsov V.E. Process modeling in Educational IT-Projects. CSIT'2008: Proceedings of the 10-rd International Workshop on Computer Science and Information Technologies (Antalya, Turkey, September 15-17, 2008). – Ufa: Ufa State Aviation Technical University, 2008. – v. 1. – pp. 227-230.
7. Рамбо Дж., Блаха М. UML 2.0. Объектно-ориентированное моделирование и разработка [Текст] / Дж. Рамбо, М. Блаха. – 2-е изд. – СПб.: Питер, 2007. – 544 с.: ил.